HyperTransport Error Management

Rick Spratt
Engineering Manager
Sun Microsystems, Inc.
Richard.Spratt@sun.com

Platform Conference
Direction • Design • Perspective • Analysis

Why Error Management

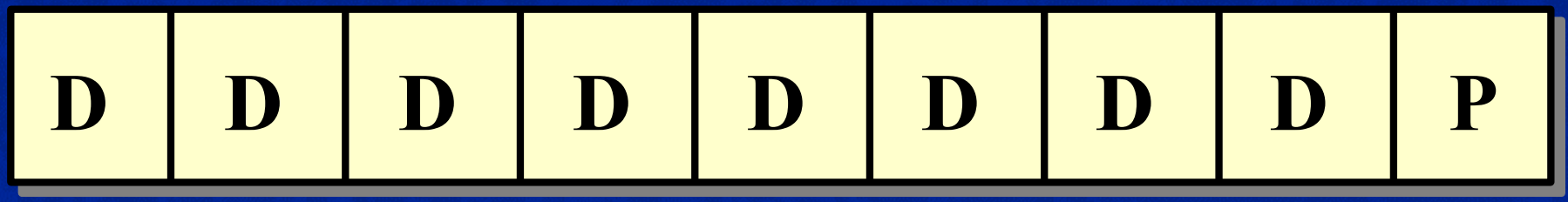
- Data integrity
  - Who wants faulty data?
- Errors are a function of:
  - Physical medium used.
  - Frequency of information transferred.

Why Error Management

- Periodic CRC checking is already in HyperTransport now.
- Error Management adds robust error recovery for high availability.
- Error Management is an extension to the current specification.

Error Protection Schemes

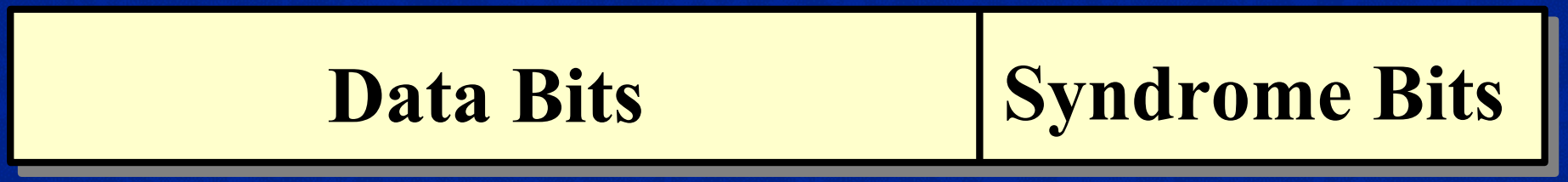
Parity:



A Parity Bit(P) is generated such that the sum of the 1's in the Data Bits(D) and the Parity Bit is always odd or even.

Error Protection Schemes

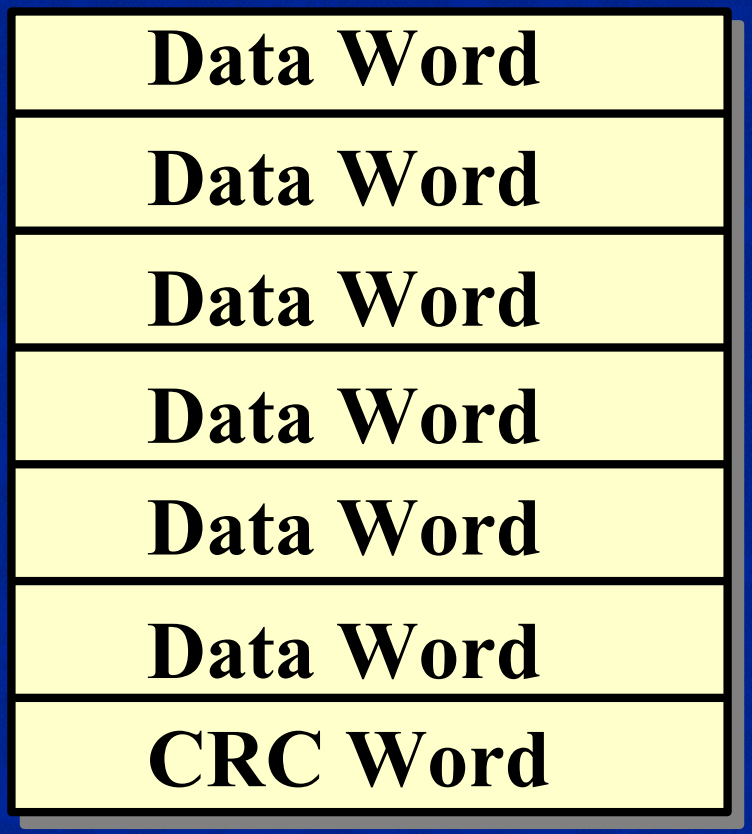
ECC:



A set of Syndrome Bits are generated such that if a single bit error occurs it can be detected and corrected. This also allows for the detection of multi-bit errors.

Error Protection Schemes

CRC:



A CRC Word is generated that is the remainder of the division of the Data Words by a CRC-32 polynomial.

Error Protection Schemes

CRC:

- Each data bit has a high potential of changing the CRC result.
- Using a 32 bit width allows for a low probability of undetected errors (e.g. 32 bits gives a 1 in 2^{32} chance of failure.
- Choice of the right polynomial is key to having the strongest coverage.

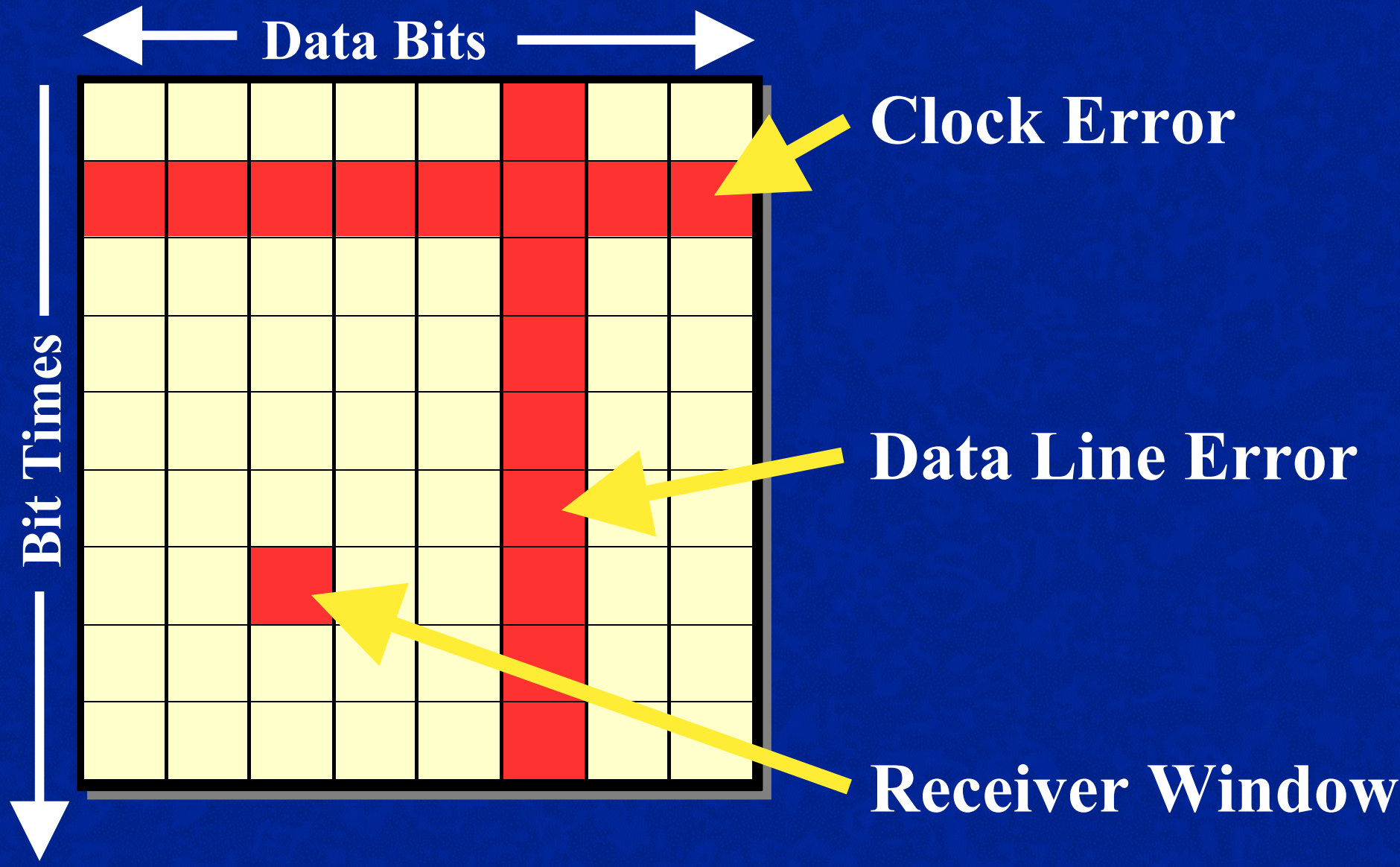
Error Protection Schemes

- Parity:
  - Good for detection of single bit errors.
- ECC:
  - Good for single bit correction and some multi-bit detection.
- CRC:
  - Good for detection of multiple bit errors.

Errors in HyperTransport

- Clock Failures
  - Takes out one or more rows
- Data Line stuck at Failures
  - Takes out a column
- Receive Window Failure
  - Takes out random bits

Errors in HyperTransport



Error Management Goals

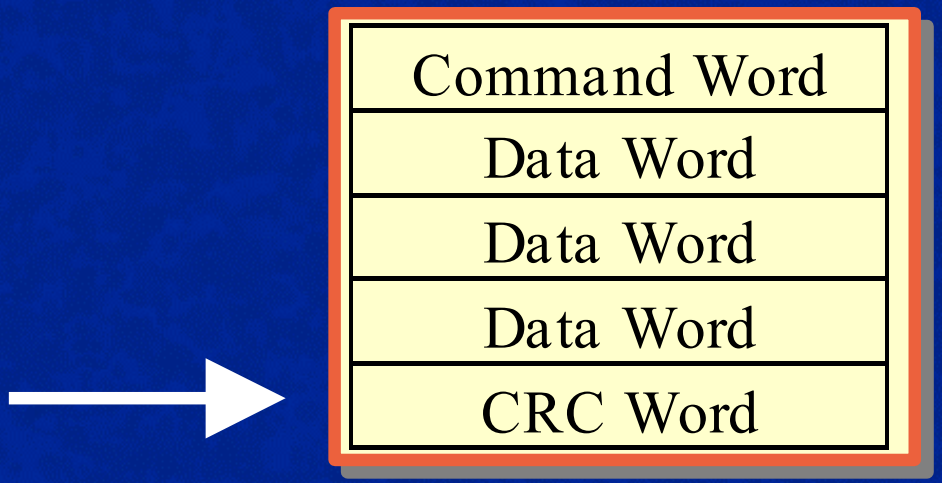
- Monitor, Detect and Recover from soft errors on HyperTransport links.
- Monitor the health of links to enable higher level fault management system to manage link failures.
- Transparent to I/O software model.
- Minimal performance impact.

Link Error Management.

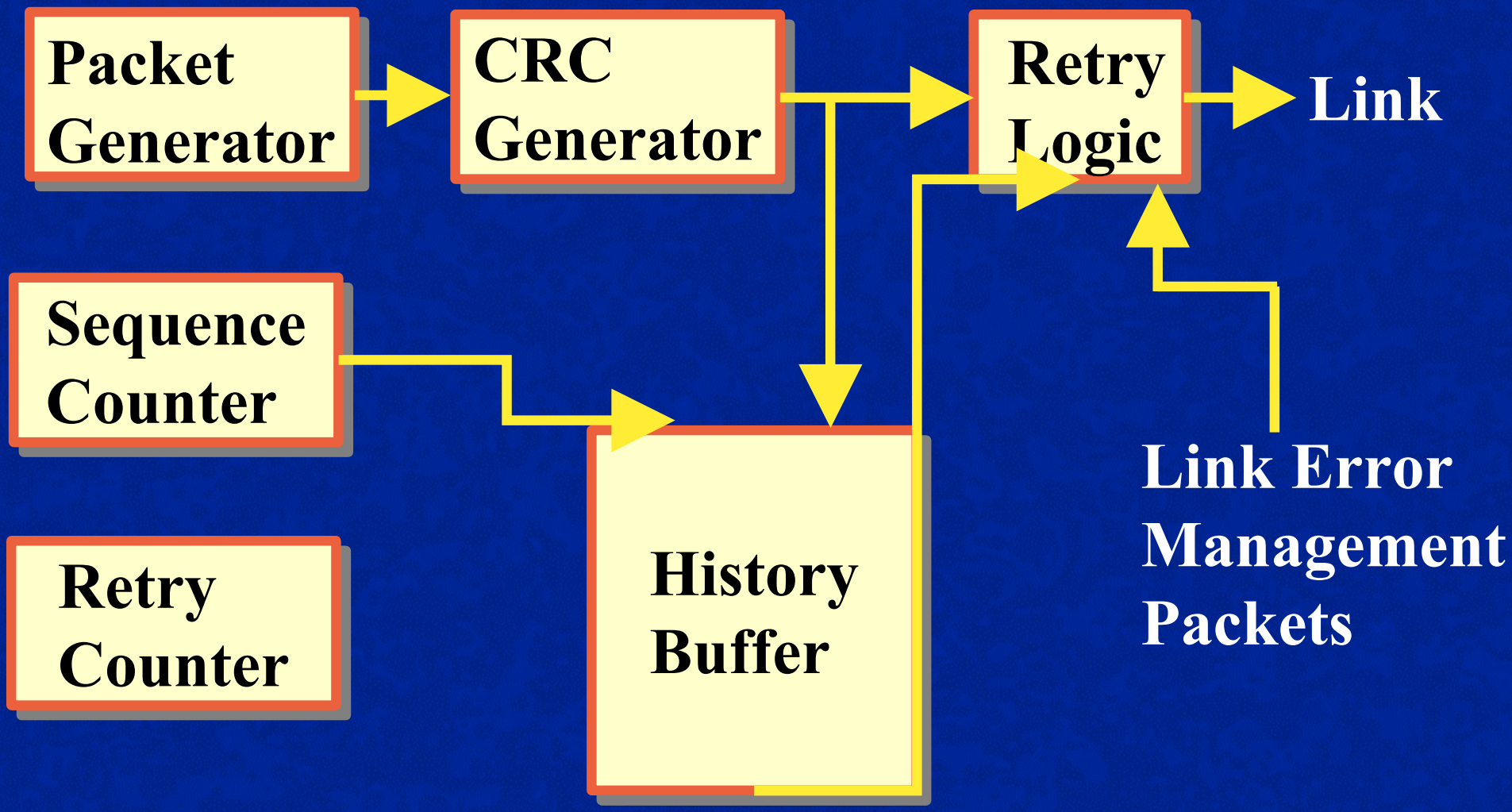
- Supplement the periodic CRC error detection mechanism.
- Add a per-packet CRC-32 to detect errors.
- Implemented on a link-by-link basis.

Packet Transmission

- Transmit side of each link appends a 4-byte CRC-32 coded word to each transmitted packet.



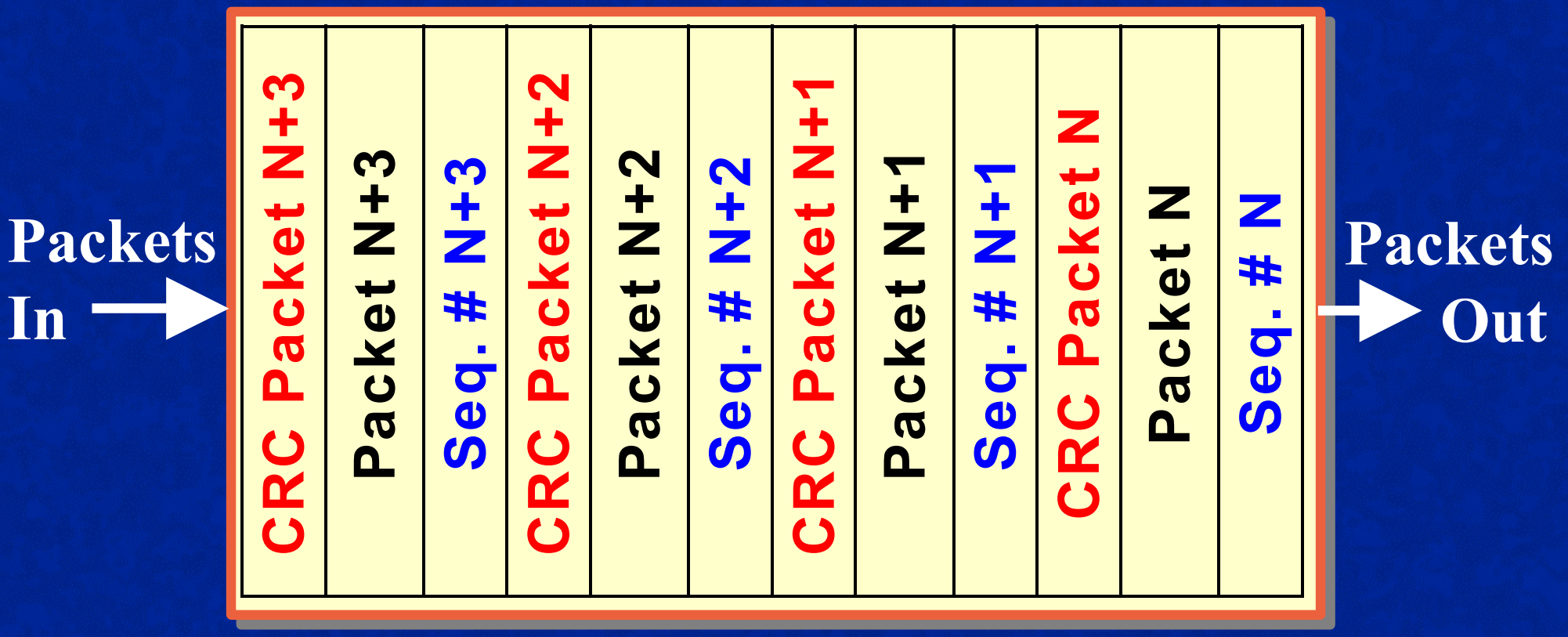
Transmitter Error Logic



Packet Sequence Number

- Saves a copy of the packet along with its sequence counter value, *xmit_nextPacketToAck* in a history buffer each time a packet is transmitted, and increments the sequence counter.

History Buffer



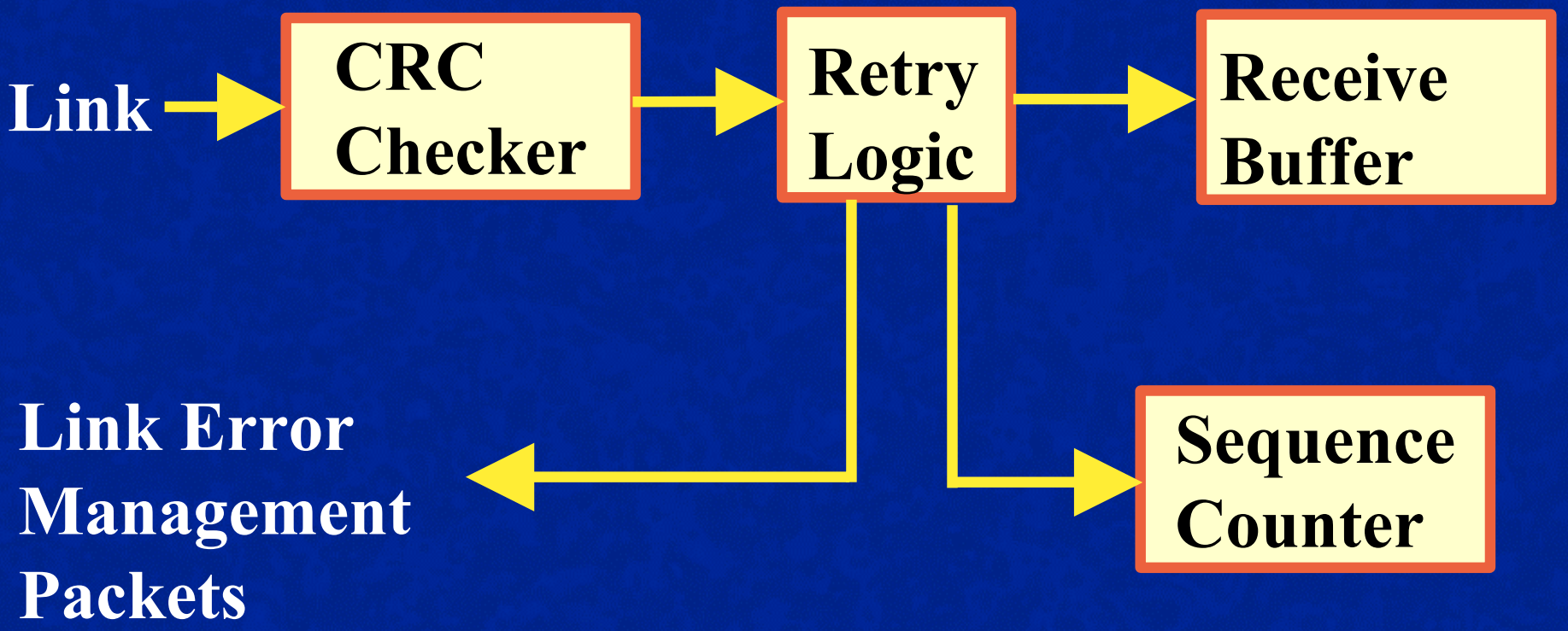
Packet Reception

- Receiver checks the CRC on each received packet.
- Increments its sequence counter value, *rcv_nextPacketToAck* each time a packet is received error-free, and increments the sequence counter.

Receiver Packet Sequence

- *The receiver includes the value of its rcv_nextPacketToAck* every time it sends an info packet to its transmitter to update buffer credits.
- *Acknowledges all packets, including rcv_nextPacketToAck* to the transmitter.

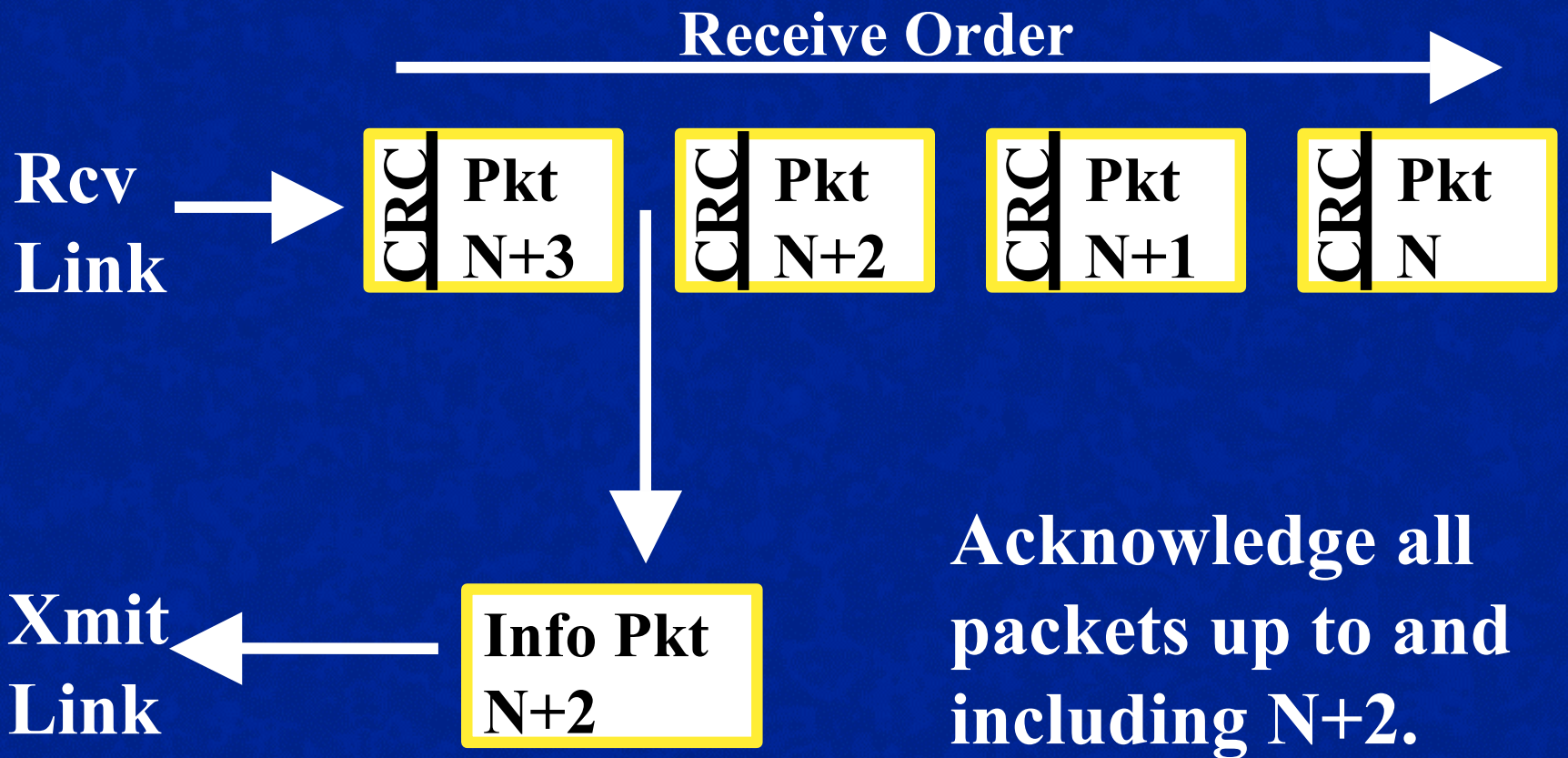
Receiver Error Logic



Packet Acknowledgement

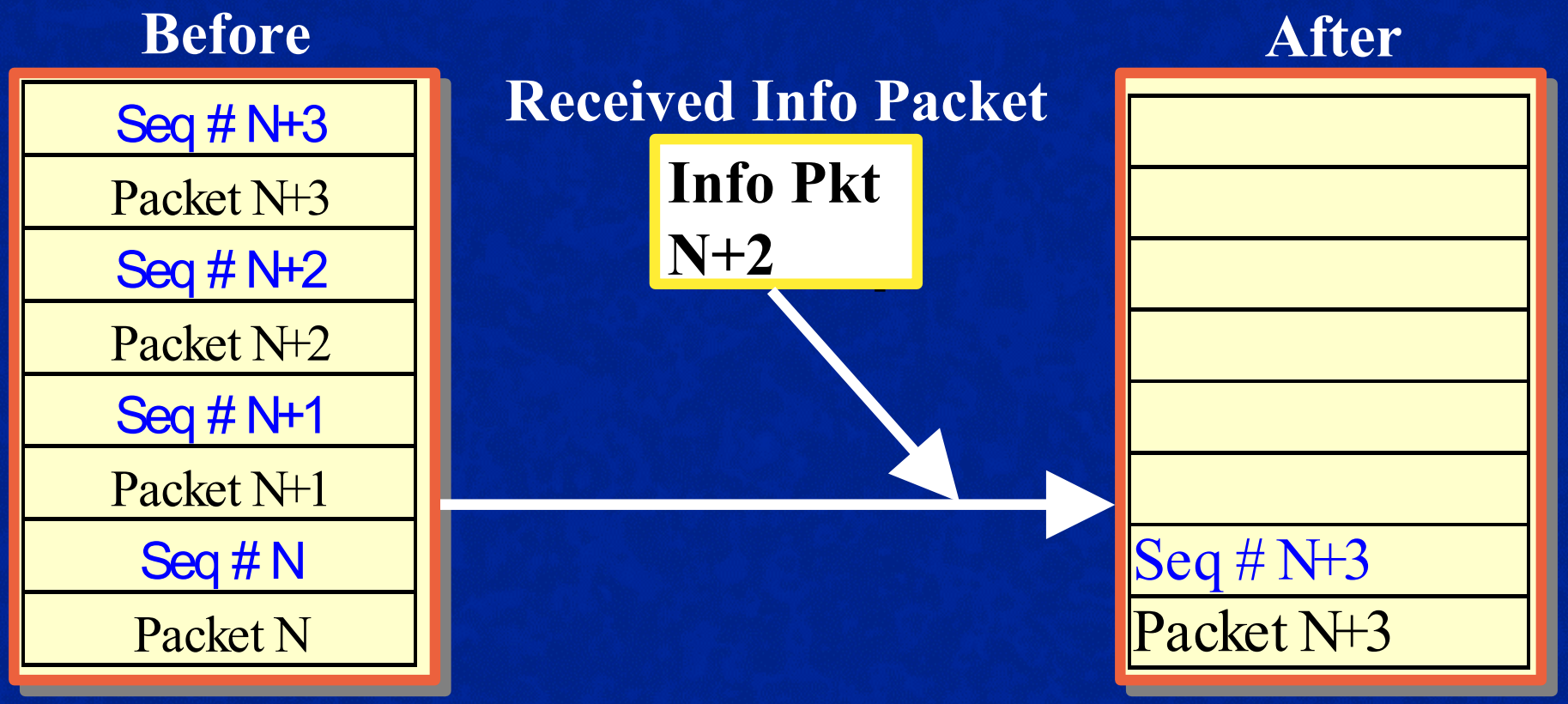
- *The packet is removed from the history buffer when it receives an acknowledgment from the receiver.*
- *A transmitter may support an implementation specific number of packets in its history buffer.*
- *The transmitter stalls if the history buffer is full.*

Receiver Packet ACK



Acknowledge all packets up to and including N+2.

Transmitter Packet ACK

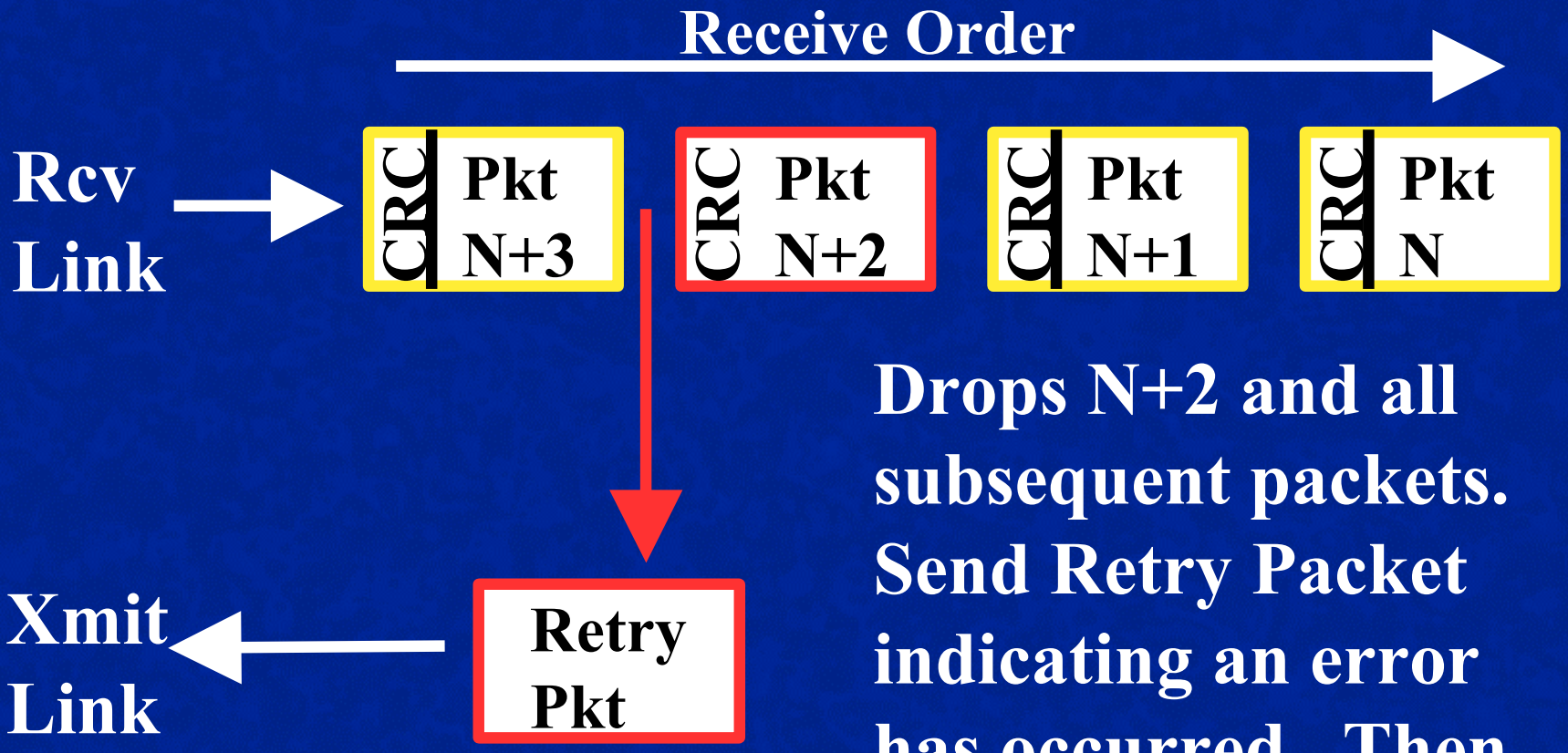


Acknowledges all packets up to and including N+2.

Packet Received in Error.

- The receiver may only execute or forward an error-free packet.
- If a packet is received in error, that packet, and all subsequent received packets are dropped.
- A special retry_info packet is sent back to the transmitter.

ACK at Receiver with Error



Drops N+2 and all subsequent packets. Send Retry Packet indicating an error has occurred. Then receiver initiates a link sync sequence.

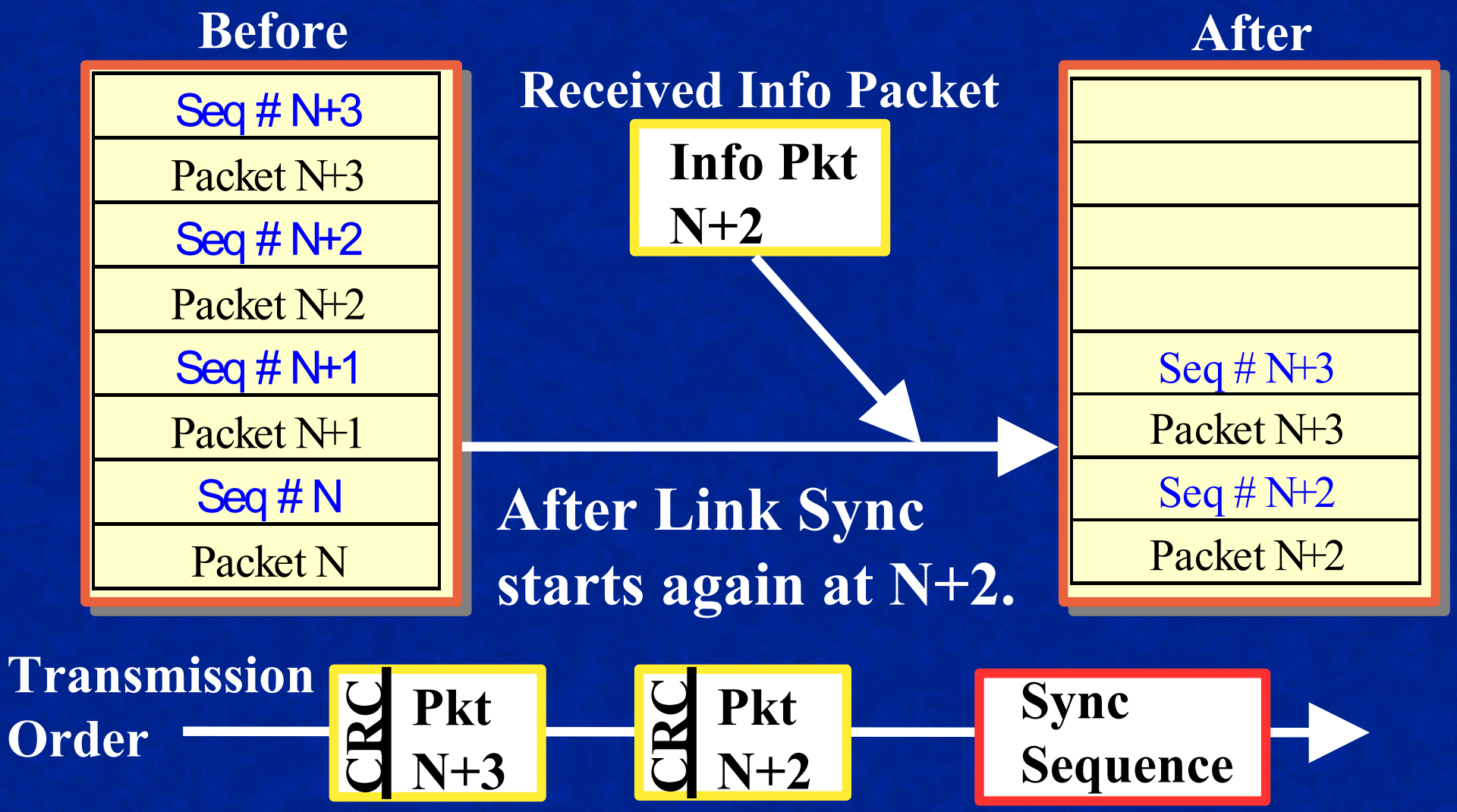
Link Synchronization

- The transmitter goes through a link synchronization sequence when it sees a retry packet and receiver link synchronization sequence.
- The receiver waits for termination of the synchronization sequence.

Retry Packets

- The receiver sends an info packet with *rcv_nextPacketToAck.*
- Transmitter receives the info packet and replays transactions from its history buffer starting from *rcv_nextPacketToAck*.
- Guarantees packet order, and that no packet is executed more than once.

Replay from History Buffer



Unrecoverable Error.

- Transmitter receives another retry with the same *rcv_nextPacketToAck* before any other intervening non-retry info packets will declare an unrecoverable error.
- The transmitter will sync-flood the HyperTransport chain, causing the host bridge to terminate all outstanding non-posted transactions in error.

Soft Error Rate Counters.

- Each transmitter maintains a Retry Counter that is incremented on each retry sequence.
- This counter should be periodically read and reset by fault management system.
- Retry count that exceeds a predefined, system dependent value, can trigger a error management/containment process.

Questions?

Where to get more Information:

- Protocol Spec Draft rev 1.03
- http://www.hypertransport.org